SMART WATER FOUNTAINS

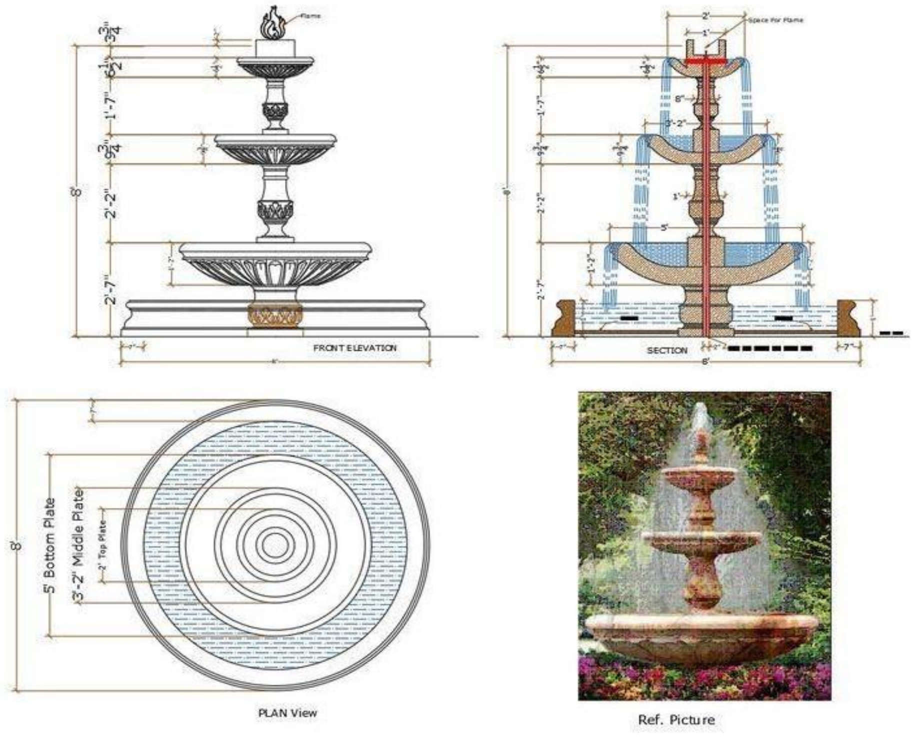
IOT PHASE-5

OBJECTIVE:

The objective of this project is to create a Smart Water Fountains system that leverages Internet of Things (IoT) technology to monitor and manage water resources efficiently. The project aims to reduce water wastage, prevent leaks, and ensure the sustainable use of water in urban and rural environments.

CONCEPT OF SMART WATER FOUNTAIN:

A smart water fountain is a drinking fountain that uses sensors and software to collect and analyze data about its usage,water quality, and other factors. This data can be used to improve the fountain's efficiency, reduce water waste, and ensure that users have access to clean, safe drinking water.



SOFTWARE AND WEB DEVELOPMENT:

1. Set up IOT Platform:

Choose an IOT platform (e.g., Thing Speak, ubidots, or AWS IOT) for data management and control.

2. Connect Microcontroller to Wi-Fi:

Write code to connect the microcontroller to your Wi-Fi network. Use arduino IDE or similar tools for programming.

3. Water Level Monitoring:

Create code on the microcontroller to read the water level sensor data and send it to the IOT platform. You can use HTTP requests or MQTT for this.

4. Controlling the Fountain:

Develop code to receive commands from the IOT platform to control the solenoid valve. This will allow users to turn the fountain on or off remotely.

5. User Interface:

Develop a web interface for users to monitor the water level and control the fountain. HTML, CSS, and JavaScript can be used for this.Use JavaScript to interact with the IOT platform's API to fetch data and send control commands.

6. Data Visualization:

Utilize web development technologies and libraries like D3.js or Chart.js to create graphs and charts to visualize the water level over time.

7. Mobile App:

If you want to provide a mobile app for controlling the fountain, consider using React Native, Flutter, or a similar framework for cross-platform development.

8. Notifications:

Implement push notifications in your web application or mobile app to

alert users when the water level is low or when the
fountain is turned on/off.

9. Testing and Debugging:

Test the system thoroughly, both the hardware and the web interface. Debug any issues and ensure they are working together seamlessly.

10. Deployment:

Once everything works as expected, deploy your web application and make it accessible to users. You can use cloud hosting services like AWS.

11. User Documentation:

Provide clear instructions for users on how to access and use the web interface or mobile app.

12. Security:

Implement proper security measures to protect the IOT system and user data, such as secure API communication and user authentication.

Remember to document the entire project and encourage students to experiment and expand on the system by adding features or improvingexisting ones. This project provides hands-on experience in both hardware and web development, making it a valuable learning experience for students.

WOKWI:

Wokwi is a versatile online platform that allows you to design, simulate, and test electronic circuits in a virtual environment.

WOKWi
Simulate IoT Projects in Your Browser
Discord Community
LinkedIn Group
Simulate with Wokwi Online
Arduino (Uno, Mega, Nano)
ESP32
STM32
Pi Pico

COMPONENTS REQUIRED:

- Raspberry Pi
- Ultrasonic Sensor
- Water Pump
- LED
- Resistor

The components required for this smart water fountain project, where the led blinks when the water level surpasses 200 cm, include a raspberry pi for control, an ultrasonic sensor to measure water levels, a water pump to control water flow, an led for visual indication, a resistor for led operation, a breadboard for prototyping, and jumper wires to establish electrical connections, all working together to create an interactive and dynamic water fountain system that autonomously adapts to changing water levels.

WIRING CONNECTIONS:

1. Ultrasonic Sensor:

Purpose: The ultrasonic sensor is used to measure water levels in the fountain.

- Connect the VCC (power) pin to the 5V output of the Raspberry Pi.
- Connect the GND (ground) pin to a GND (ground) pin on the Raspberry Pi.
- Connect the TRIG (trigger) pin to GPIO pin 17 on the Raspberry Pi.
- Connect the ECHO (echo) pin to GPIO pin 18 on the Raspberry Pi.

2. Water Pump:

Purpose: The water pump controls the flow of water within the fountain.

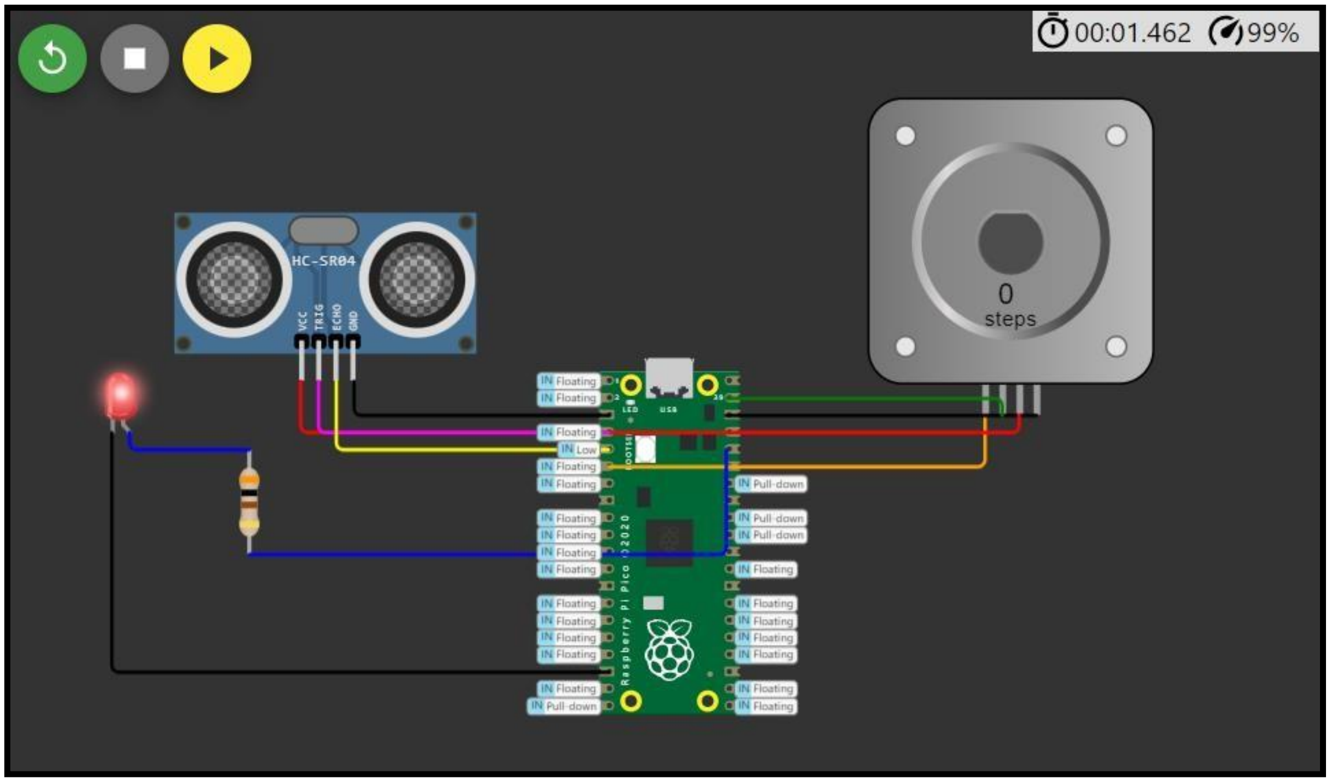
- Connect the positive (red) wire of the water pump to an external power supply suitable for the pump's voltage and current requirements.
- Connect the negative (black) wire of the water pump to the collector (C) of an NPN transistor or use a motor driver module to control the pump.
- Connect the emitter (E) of the transistor to the GND (ground) of the Raspberry Pi.
- Connect the base (B) of the transistor to GPIO pin 4 on the Raspberry Pi through a current-limiting resistor (220-330 ohms).

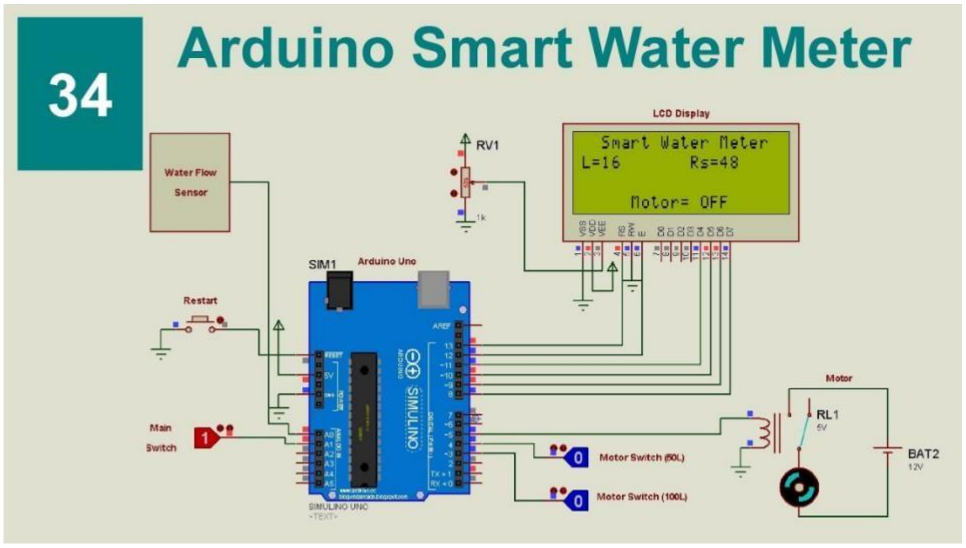
3. LED (with Resistor):

Purpose: The LED serves as a visual indicator of the water level.

- Connect the longer leg (anode) of the LED to a current-limiting resistor (220-330 ohms).
- Connect the other end of the resistor to GPIO pin 5 on the Raspberry Pi.
- Connect the shorter leg (cathode) of the LED directly to a GND (ground) pin on the Raspberry Pi.

These connections are essential for the smart water fountain project, allowing the Raspberry Pi to interface with the ultrasonic sensor, water pump, and LED to create a dynamic and interactive water feature.





CODE DESCRIPTION:

```
import time

TRIG_PIN =

2

ECHO_PIN=3

PUMP_PIN =4

LED_PIN = 5

ultrasonic_sensor=

Ultrasonic(TRIG_PIN,CHO_PIN) pump =

Motor(PUMP_PIN)



led = LED(LED_PIN

while True:

   distance =

   ultrasonic_sensor.distance_cm if
```

```
  distance > 200:
    led.blink(on_time=0.5,
    off_time=0.5) pump.on() #
    Water pump is turned on
  else:
    led.off()
    pump.off()
  time.sleep(0.1)
```

This code effectively simulates a smart water fountain project in the Wokwi environment, where the LED blinks when the water level is above 200 cm and stops when it falls below that threshold.

Platform UI code for Smart Water Fountains:

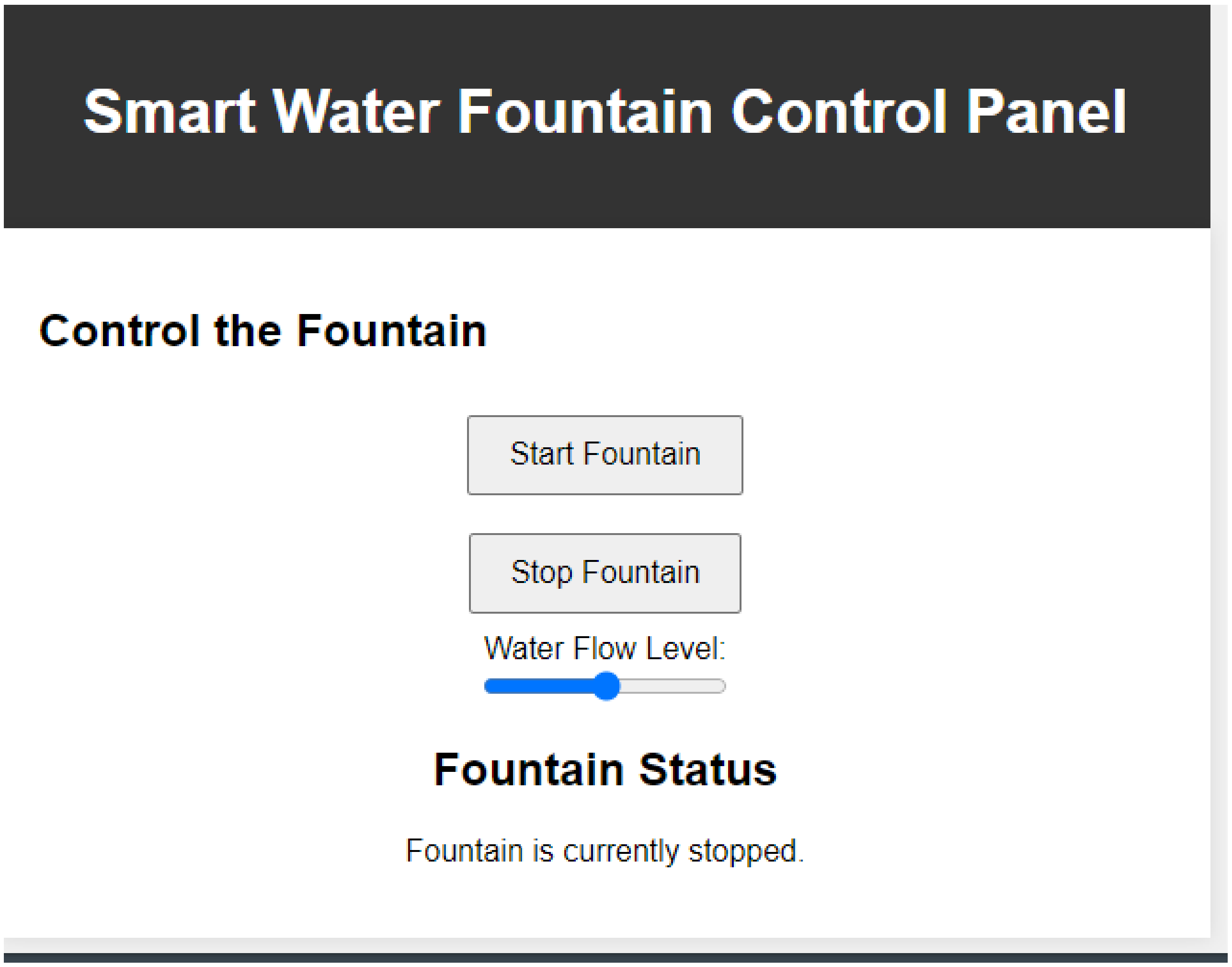
```
 <!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Smart Water Fountain Control</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
```

```html
    <header>
        <h1>Smart Water Fountain Control Panel</h1>
    </header>

    <main>
        <section class="water-fountain-control">
            <h2>Control the Fountain</h2>
            <div class="controls">
                <button id="start-fountain">Start Fountain</button>
                <button id="stop-fountain">Stop Fountain</button>
                <label for="water-flow">Water Flow Level:</label>
                <input type="range" id="water-flow" min="0" max="100" value="50">
            </div>
        </section>

        <section class="water-fountain-status">
            <h2>Fountain Status</h2>
            <p id="status-text">Fountain is currently stopped.</p>
        </section>
    </main>

    <script src="script.js"></script>
</body>
</html>
```

```
/* styles.css */

body {

   font-family: Arial, sans-serif;

   background-color: #f0f0f0;

}


header {

   background-color: #333;

   color: white;

   text-align: center;

   padding: 20px;

}


main {

   max-width: 800px;

   margin: 0 auto;

   padding: 20px;

   background-color: #fff;

   box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}


.water-fountain-control {

   margin: 20px 0;

}
```

```
.controls {
   display: flex;
   flex-direction: column;
   align-items: center;
}

button {
   padding: 10px 20px;
   margin: 10px;
   font-size: 16px;
}

.water-flow {
   margin-top: 10px;
}

.water-fountain-status {
   text-align: center;
}

/* Add more CSS for styling as needed */
// script.js
document.addEventListener("DOMContentLoaded", function () {
   const startButton = document.getElementById("start-fountain");
   const stopButton = document.getElementById("stop-fountain");
   const waterFlowRange = document.getElementById("water-flow");
   const statusText = document.getElementById("status-text");

   startButton.addEventListener("click", () => {
       // Implement code to start the fountain
       statusText.innerText = "Fountain is running.";
   });

   stopButton.addEventListener("click", () => {
       // Implement code to stop the fountain
       statusText.innerText = "Fountain is stopped.";
   });

   waterFlowRange.addEventListener("input", () => {
       const flowLevel = waterFlowRange.value;
       // Implement code to adjust water flow level
       statusText.innerText = `Water flow level: ${flowLevel}%`;
   });
});
```

OUTPUT:



Smart Water Fountain Control Panel

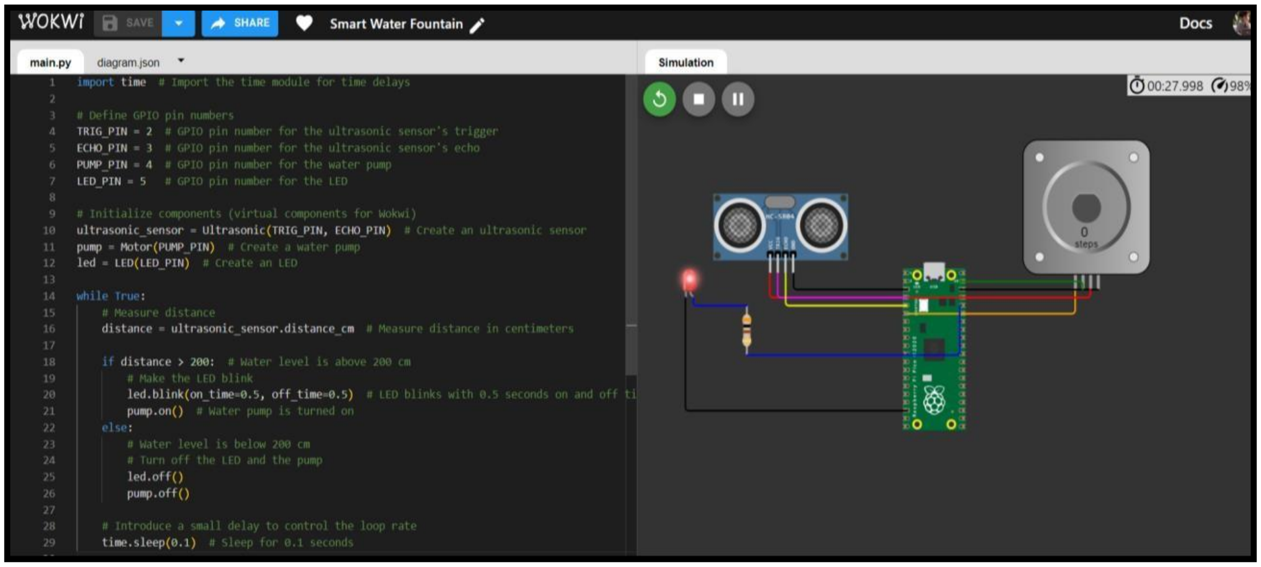
Control the Fountain

Fountain Status

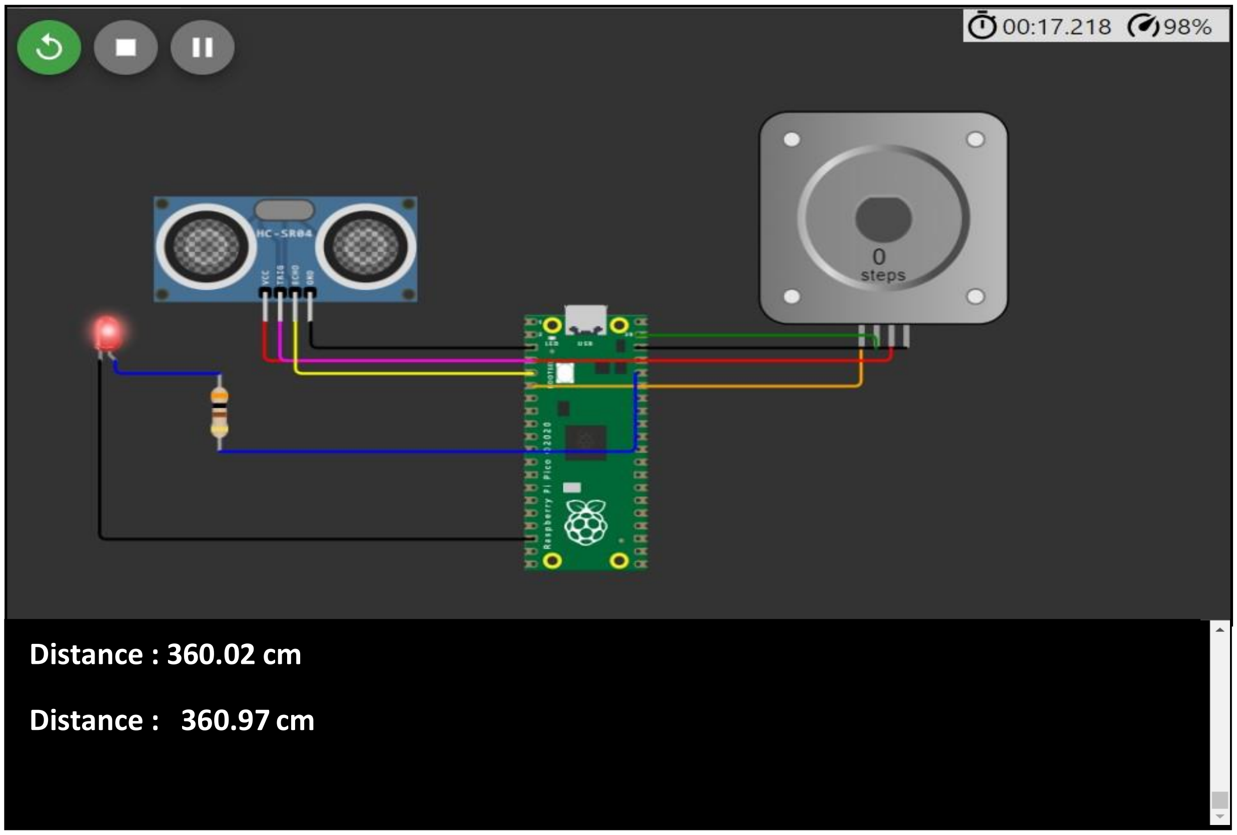
Fountain is currently stopped.

RESULT ANALYSIS:

This project aims to create a virtual smart water fountain simulation using Wokwi, integrating a Raspberry Pi (or a compatible microcontroller) with virtual components such as an ultrasonic sensor, water pump, and LED. The simulation successfully monitors water levels, causing the LED to blink when the water level surpasses 200 cm and activating the water pump accordingly. This project showcases the ability of virtual components to emulate the functionality of a real-world system within a simulated environment



```python
import time  # Import the time module for time delays

# Define GPIO pin numbers
TRIG_PIN = 2  # GPIO pin number for the ultrasonic sensor's trigger
ECHO_PIN = 3  # GPIO pin number for the ultrasonic sensor's echo
PUMP_PIN = 4  # GPIO pin number for the water pump
LED_PIN = 5   # GPIO pin number for the LED

# Initialize components (virtual components for Wokwi)
ultrasonic_sensor = Ultrasonic(TRIG_PIN, ECHO_PIN)  # Create an ultrasonic sensor
pump = Motor(PUMP_PIN)  # Create a water pump
led = LED(LED_PIN)  # Create an LED

while True:
    # Measure distance
    distance = ultrasonic_sensor.distance_cm  # Measure distance in centimeters

    if distance > 200:  # Water level is above 200 cm
        # Make the LED blink
        led.blink(on_time=0.5, off_time=0.5)  # LED blinks with 0.5 seconds on and off ti
        pump.on()  # Water pump is turned on
    else:
        # Water level is below 200 cm
        # Turn off the LED and the pump
        led.off()
        pump.off()

    # Introduce a small delay to control the loop rate
    time.sleep(0.1)  # Sleep for 0.1 seconds
```



```
Distance : 360.02 cm

Distance :   360.97 cm
```

CONCLUSION:

This Wokwi-based smart water fountain simulation project effectively illustrates the control and monitoring of water levels using a Raspberry Pi and virtual components. The project demonstrated the practicality of using Wokwi's virtual environment for hardware simulation, allowing precise testing and visualization of system functionality without physical components. The LED and water pump responded to water level changes as expected, showcasing the potential for virtual hardware modeling.